RFC 6365 : Terminology Used in Internationalization in the IETF

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 septembre 2011

Date de publication du RFC : Septembre 2011

https://www.bortzmeyer.org/6365.html

L'internationalisation des protocoles développés par l'IETF est depuis quinze ans un des sujets chauds de cette organisation. Réussir à rendre tous les protocoles Internet adaptés aux différentes langues, écritures et cultures nécessite d'avoir un socle linguistique commun et c'est le rôle de ce RFC, qui décrit la terminologie en usage à l'IETF. Il remplace le RFC 3536 [1] dans le rôle du texte « l'internationalisation pour les débutants ». Le texte est long car le sujet est complexe.

Cette terminologie est d'autant plus importante que les discussions sur le sujet sont souvent victimes d'un vocabulaire flou et mal utilisé. Ainsi, on voit parfois apparaître le terme de « caractère latin » sans que celui-ci soit bien défini. Seulement ASCII ou bien également les caractères composés, utilisés par exemple en français ? Autre exemple, à l'ICANN, les discussions sur les IDN sont souvent rendues incompréhensibles par une confusion fréquente entre des concepts aussi différents que « langue » et « écriture ». Si vous lisez dans un article ou un rapport des termes absurdes comme « noms de domaines multilingues », faites lire ce RFC 6365 à l'auteur de ce texte, il en a besoin...

Ce RFC est en anglais et fixe une terminologie anglophone. Dans mon article, je vais donner le terme anglais défini et la traduction française que j'utilise (sans consulter les normes officielles, qui ne sont pas toujours de grande qualité et souvent faites par des gens qui ignorent le domaine). Mais le lecteur doit bien être conscient du fait qu'il n'existe pas forcément de terminologie standard en français. Même en anglais, le consensus est loin d'exister sur tous ces termes, et le caractère ultra-sensible des questions de langue n'arrange rien.

Petit rappel de l'objectif poursuivi, en section 1 : comme le dit le RFC 2277, document fondateur, l'internationalisation concerne les humains, pas les protocoles. On ne va donc pas traduire le `GET` de HTTP.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3536.txt`

Par contre, lorsqu'un protocole manipule des chaînes de caractères qui vont être lues ou écrites par des humains, il faut que cela soit possible quelle que soit la langue ou l'écriture utilisée. Cette politique de l'IETF est donc différente de celle, qu'on entend souvent à l'ICANN, comme quoi tout le monde n'a qu'à parler anglais, langue internationale de toute façon.

Avoir un vocabulaire commun est évidemment essentiel pour les discussions au sein de l'IETF, et pour la qualité et l'homogénéité des documents produits (les RFC). Ce RFC 6365 sert aussi de tutoriel sur l'internationalisation en expliquant les concepts de base aux membres de l'IETF, un public d'ingénieurs qui n'est pas forcément très sensible à cette question. Notre RFC rappelle ainsi que, dans beaucoup de groupes de travail de l'IETF, lorsque quelqu'un soulève la question de l'internationalisation du protocole en cours de développement, les réponses « faut-il vraiment s'en préoccuper ? » sont souvent nombreuses. Il y a donc un certain fossé à combler.

Voyons maintenant les définitions générales (section 2). Le RFC ne pouvait pas tout couvrir et un des plus grands débats dans le groupe avait été l'étendue de la liste des termes à définir. Le choix a été fait de se restreindre, pour augmenter la probabilité que les non-spécialistes de l'internationalisation lisent le texte. Je ne reprends ici qu'une partie des termes, ceux que je trouve indispensables, ou bien dont la définition est mal connue.

Une **langue** ("*language*") est une façon de communiquer entre humains, notamment par la parole ou l'écriture. Lorsqu'une forme écrite et une forme parlée existent pour une langue, elles sont parfois très proches... et parfois très distantes. Ce terme n'inclus **pas** les langages de programmation.

Une **écriture** ("*script*") est un ensemble de caractères graphiques qui sont utilisés pour la forme écrite d'une ou de plusieurs langues. C'est le cas de l'écriture latine, cyrillique, arabe (attention dans ce cas, arabe est également le nom d'une langue) et han. L'IETF s'occupant surtout de l'écrit (la langue parlée peut être transmise par des protocoles comme RTP mais elle n'est typiquement pas interprétée par ces protocoles), presque toujours, les protocoles IETF s'occupent d'écritures, pas de langues. C'est ainsi que les IDN sont des noms de domaines multi-écritures et surtout pas multilingues. (Le RFC note explicitement que « multilingue » est un de ces termes flous et passe-partout, qu'il faut éviter.)

(La définition de l'internationalisation dans notre RFC est d'ailleurs modeste, « permettre ou améliorer l'utilisation de caractères non-ASCII dans les protocoles IETF », par rapport celle du W3C, plus ambitieuse, « permettre l'adaptation facile à n'importe quelle langue, écriture ou culture ».)

La relation entre les langues et les écritures est complexe : il y a beaucoup moins d'écritures que de langues et la plupart des écritures peuvent servir à plusieurs langues. D'autre part, si certaines langues n'ont jamais été écrites qu'avec une seule écriture (le français avec l'écriture latine, le russe avec la cyrillique), d'autres ont changé, comme le turc ou le vietnamien. L'effondrement de l'URSS a eu entre autres pour conséquence l'abandon du cyrillique par plusieurs langues comme le mongol. Autre exemple, le malais est passé récemment (et la transition n'est pas terminée) du jawi à l'alphabet latin.

On voit parfois le terme **alphabet** pour désigner une écriture mais, comme toutes ne sont pas alphabétiques (cf. section 4.1), ce terme est plutôt à éviter.

Un **système d'écriture** ("*writing system*") est un ensemble de règles pour utiliser une écriture dans le cas d'un langage donné. Ainsi, le français et l'anglais utilisent la même écriture, mais avec des règles différentes (par exemple pour la ponctuation).

Un **caractère** (”*character*”) est un membre de l'ensemble des éléments qui servent à représenter les données écrites. Le terme est plus flou qu'il n'y parait car il existe des tas de cas qui défient la classification comme le digramme « ll » du catalan. En anglais, les informaticiens disent souvent ”*char*” au lieu de ”*character*”, vu que c'est le type de données utilisé dans plusieurs langages de programmation. En tout cas, un caractère n'est **pas** une représentation graphique, un **glyphe**. Par exemple, il n'y a qu'un seul caractère A alors qu'il en existe de nombreuses représentations possibles (il y a un excellent article sur ce sujet dans le livre de Hofstadter, « ”*Metamagical Themas*” »).

En parlant de glyphe, sa définition dans le RFC est « l'image d'un caractère rendue sur une surface ». On aura donc un glyphe différent si on affiche avec une résolution différente. Unicode a une autre définition, basée plutôt sur l'encodage d'une police particulière, et où le glyphe est donc le même quel que soit le périphérique de sortie. Dans la définition d'Unicode, chaque glyphe a un **code de glyphe**, qui est un index dans une police donnée (et qui n'est pas standard : deux auteurs de polices peuvent utiliser des codes de glyphes différents).

Un **jeu de caractères** (”*repertoire*”) est simplement un ensemble de caractères, rigoureusement défini (c'est par exemple à ce stade qu'on décide si œ est un seul caractère ou deux).

Un **jeu de caractères codé** (”*Coded Character Set*”, CCS) est un ensemble de règles qui définissent un jeu de caractères **et** une représentation codée (typiquement, par un nombre entier unique par caractère). Unicode ou ISO-8859-6 définissent un CCS.

L'**encodage** (”*character encoding scheme*”) est la représentation en machine des caractères, c'est-à-dire la définition de quelle chaîne de bits représente tel caractère. Si des normes comme ISO 8859 ne définissent qu'un seul encodage possible, Unicode en a plusieurs (le plus connu étant UTF-8, qui est le protocole par défaut dans l'Internet, cf. RFC 2277).

Le **transcodage** (”*transcoding*”) est la traduction d'un encodage dans un autre, comme le font les outils Unix iconv ou recode.

Je laisse en anglais le terme ”*charset*” car il est purement technique et n'est **pas** la même chose qu'un jeu de caractères tel que défini plus haut. Ce terme de ”*charset*” est très utilisé dans les normes IETF (notamment MIME) et les ”*charsets*” peuvent être enregistrés dans un registre IANA `<https://www.iana.org/assignments/character-sets>` selon les règles du RFC 2978. Un ”*charset*” est un moyen de passer d'une séquence d'octets à une chaîne de caractères abstraits. C'est donc la combinaison d'un jeu de caractères codé et d'un encodage. ”*charset*” doit être considéré comme un mot-clé, à utiliser sans regarder de trop près son étymologie, et le RFC déconseille notamment de le remplacer par ”*character set*”, qui peut avoir d'autres définitions en anglais. On va donc dire que ”*charset*” est un néologisme spécifique à l'IETF.

La section suivante, la 3, liste les organismes de normalisation qui ont à voir avec l'internationalisation (ou plutôt une partie d'entre eux). Par exemple, l'ISO a une norme de jeu de caractères, ISO 10646, qui fait même partie des rares normes ISO disponibles gratuitement en ligne (dans sa version anglaise, uniquement, et après acceptation d'un long texte juridique menaçant, c'est pour cela que je ne mets pas de lien ici). Cette norme est synchronisée avec Unicode et il n'y a donc pas de raison de la lire, Unicode offre davantage de choses. Le groupe de travail SC2/WG2 qui produit la norme ISO a un site Web `<http://std.dkuug.dk/JTC1/SC2/WG2/>` qui donne des aperçus du travail en cours.

D'autres normes ISO ont un rapport direct avec l'internationalisation comme ISO 639 qui gère un catalogue de langues, avec des codes alphabétiques (comme `fr` pour le français) ou ISO 3166 qui gère

une liste de pays, avec leurs codes (comme `fr` pour la France). Ces normes forment la base des étiquettes de langue (cf. RFC 5646), utilisées par exemple dans le Web pour indiquer la langue d'une page.

Le consortium Unicode est un autre organisme de normalisation, qui gère le jeu de caractères du même nom, ainsi qu'un certain nombre de travaux qui lui sont liés, comme la définition d'expressions rationnelles pour Unicode. On l'a vu, Unicode et ISO 10646 sont synchronisées et, techniquement, on peut se référer à l'une ou l'autre norme, sans conséquences pratiques. Ce RFC 6535 ne le dit pas mais c'est pour des raisons politiciennes qu'ISO 10646 était plus souvent cité, l'ISO, organisation privée, tout comme le consortium Unicode, étant parfois considérée par certains comme plus ouverte (alors que l'essentiel de ses documents n'est pas distribué en ligne).

Le Web est une des applications où l'internationalisation est la plus avancée, ayant été prise en compte pratiquement dès le début. C'est le rôle du W3C, qui gère des normes comme XML (dont le modèle de caractères est Unicode depuis la première version).

Enfin, et c'est important dans le contexte de l'internationalisation, il existe aussi un certain nombres d'organismes nationaux qui ont parfois une activité touchant à ce sujet.

Comme le cœur de l'internationalisation des protocoles est la définition d'un jeu de caractères commun, la section 3.2 liste le vocabulaire spécifique d'Unicode. Il faut par exemple savoir ce que sont UCS-2, UTF-16 et UCS-4/UTF-32, que le BMP ("*Basic Multilingual Plane*") est composé des 65 536 premiers caractères d'Unicode (qui sont traités à part par certains encodages), qu'UTF-8, normalisé dans le RFC 3629, est l'encodage recommandé pour tous les protocoles IETF, mais que l'IETF a un encodage spécifique pour les noms de domaines, Punycode, normalisé dans le RFC 3492. Encodant n'importe quel chaîne Unicode avec uniquement des caractères ASCII, Punycode peut sembler une solution tentante au problème d'internationalisation d'un vieux protocole, mais il vient aussi avec ses propres problèmes (cf. RFC 6055).

On l'a dit, il existe souvent des SDO nationales qui, avant l'avènement d'Unicode, ont souvent développé des jeux de caractères locaux, permettant de représenter les écritures utilisées dans leur pays. La section 3.3 rappelle ainsi que ces jeux de caractères sont encore souvent très utilisés, comme Shift-JIS pour le japonais.

Ce sont ces jeux de caractères, et leurs encodages (dans la plupart des définitions, le jeu de caractères et l'encodage sont intrinsèquement liés ; Unicode est une exception), qui sont enregistrés sous le nom de "*charsets*" à l'IANA, dans `<https://www.iana.org/assignments/character-sets>`. Cette liste n'est pas connue pour sa grande rigueur : beaucoup de noms ont été ajoutés au pifomètre. Elle ne s'appuie pas toujours sur des normes reconnues.

Le plus connu des "*charsets*" est évidemment US-ASCII, historiquement le jeu par défaut de l'Internet, depuis le RFC 20. Sa domination a officiellement pris fin lors de la sortie du RFC 2277, qui le remplaçait comme "*charset*" par défaut par son sur-ensemble UTF-8. À noter que la norme ASCII originale était uniquement un jeu de caractères, et que le terme « ASCII » utilisé à l'IETF lui ajoute un encodage spécifique aux protocoles Internet (champ de huit bits, le bit de plus fort poids toujours à zéro, les sept autres stockant le code ASCII) qui n'a été formellement décrit qu'avec le RFC 5198, bien plus tard...

Ces histoires de caractères font surgir plein d'autres problèmes de terminologie. D'où la section 4, qui lui est consacrée. Ainsi, un **point de code** ("*code point*") est le numéro que la plupart des normes attribuent à chaque caractère, pour le référencer sans ambiguité. C'est en général un nombre entier strictement positif mais, dans la norme ASCII originale, c'était un couple de nombres, identifiant la ligne et

la colonne d'une table. « A » était ainsi 4/1 (ligne 4, colonne 1), en ASCII, alors qu'il est 65 en décimal et U+0041 en Unicode.

Et connaissez-vous les **caractères combinants** ("*combining characters*") ? Il s'agit d'une particularité d'Unicode, qui met en cause la définition typographique de « caractère » puisqu'un caractère combinant se... combine avec son prédécesseur (par exemple, U+0301 est l'accent aigu combinant, donc U+0065 suivi de U+0301 fait un « é »).

Autre concept important pour les chaînes de caractères, la **canonicalisation** ("*normalization*" ; notez que, en anglais, « "*canonicalization*" a un sens très précis dans Unicode, et distinct de celui de "*normalization*", qui justifie l'emploi d'un mot séparé). C'est le processus par lequel une chaîne de caractères est mise sous une forme canonique, permettant les comparaisons (autrement, comparer U+00E9 avec U+0065 U+0301 pourrait échouer alors que, dans les deux cas, il s'agit d'un é).

Quant à la **casse** ("*case*"), c'est le fait pour un caractère d'exister en deux formes, une majuscule et une minuscule. Souvent, la correspondance entre les deux formes est un-un (chaque minuscule a une et une seule majuscule, et réciproquement) mais ce n'est pas toujours le cas. Pire, le changement de casse peut dépendre de la langue. Cette dépendance est mise en œuvre, par exemple, en Java et, outre qu'elle est très coûteuse en ressources, elle produit forcément des résultats différents selon la "*locale*" (voir section 8 pour ce terme).

Enfin, place au **tri** ("*sorting*") et au **classement** ("*collation*"). Le classement est le fait de d'ordonner les caractères. C'est un sujet très vaste. Même en ASCII, il existe au moins deux classements distincts (un par les numéros, A, B, C, D..., a, b, c, et un selon l'ordre alphabétique, A, a, B, b, C, c...). En ISO 8859-1, le cas des caractères composés n'est pas traité de manière uniforme car le classement dépend de la langue (comparez un annuaire téléphonique en France et en Suède pour voir où sont placés les noms propres commençant par un caractère composé). Sans connaître la langue, on ne peut pas espérer classer les caractères Unicode d'une manière qui ne surprenne pas l'utilisateur. Quant au tri, c'est le processus effectif de mise des caractères dans l'ordre du classement. Il fait l'objet d'innombrables algorithmes qui ont fait la joie et le malheur de centaines de milliers d'étudiants en programmation.

Même les adjectifs utilisés pour les différents types de caractères ne sont pas toujours connus. La section 4.1 explique par exemple que le terme d'idéogramme ne décrit pas vraiment correctement les écritures comme la chinoise. Si U+260E est bien une icône ([Caractère Unicode non montré [2]], un téléphone), beaucoup de caractères chinois sont utilises phonétiquement, sans coder une idée.

En tout cas, en français, une chose est sûre, on ne devrait pas parler de « caractère accentué » pour des signes comme ç qui n'ont pas d'accent. Le terme Unicode de diacritique ("*diacritic*") convient mieux. Et, évidemment, il ne s'agit pas de caractères spéciaux <https://www.bortzmeyer.org/pas-de-caracteres-spec
html>.

Unicode soulève un autre problème intéressant : la liste des caractères est très longue et il est souvent utile d'en définir des sous-ensembles. Mais attention : contrairement à ASCII ou aux ISO 8859, qui étaient statiques, Unicode grossit tout le temps (la dernière version a été la 6 <https://www.bortzmeyer.org/unicode-6-0.html>). Un sous-ensemble défini en intension comme « toutes les lettres majuscules » grossira donc lui aussi à chaque nouvelle version d'Unicode (une question qu'il a fallu résoudre pour IDN, voir le RFC 5892).

2. Car trop difficile à faire afficher par LaTeX

On l'a dit, le but de l'internationalisation est d'aider les humains, de leur rendre la vie plus facile. Il faut donc un peu se pencher sur les interfaces utilisateur, même si celles-ci sont traditionnellement hors de la portée de l'IETF (qui normalise « au dessus du câble et en dessous de l'application »). En effet, certaines caractéristiques des interfaces utilisateur peuvent affecter les protocoles. D'où l'importance d'apprendre quelques nouveaux termes.

Une **méthode d'entrée** ("*input method*") est un mécanisme qui permet d'envoyer du texte à une application. Le périphérique d'entrée est souvent un clavier mais les claviers ayant des tailles limitées, les différentes méthodes d'entrée fournissent des mécanismes pour saisir d'autres caractères. Le clavier sur lequel je suis en train de taper n'a pas de majuscules avec diacritiques donc, pour faire un É, je tape "*Compose*" (touche dite Windows, sur mon clavier) puis E puis ' (le signe sous le 4). Plus complexe, et ayant bien plus de caractères qu'il n'y a de touches sur un clavier, l'écriture chinoise peut être saisie par plusieurs méthodes différentes : taper les sons au clavier, par exemple.

Les protocoles IETF ont bien de la chance : ils n'échangent que des bits, sans se soucier de savoir comment les afficher. Un serveur HTTP, par exemple, n'a aucun besoin de connaître l'écriture arabe ou la chinoise (ou même de comprendre Unicode), il se contente d'envoyer les octets. Mais le navigateur Web, lui, va devoir afficher les textes pour l'utilisateur. Il s'appuie pour cela sur les **règles de rendu** ("*rendering rules*"), qui sont les algorithmes d'affichage du texte à l'écran. Avec certaines écritures, c'est encore relativement simple : chaque caractère est représenté par un glyphe dans une police de caractères et on pousse simplement ce glyphe vers le périphérique de sortie. Mais dans d'autres écritures, il faut modifier le caractère selon le contexte (par exemple, avec l'écriture arabe, selon qu'il est au début ou à la fin de la phrase). Et, bien sûr, une des choses les plus difficiles pour un moteur de rendu, est de gérer le bidi <https://www.bortzmeyer.org/affichage-bidi.html>. Comme je l'ai dit, les protocoles IETF n'ont pas besoin de connaître ces subtilités (un serveur HTTP envoie des textes en alphabet arabe comme ceux en alphabet latin : dans l'ordre de saisie). Mais il est bon que leurs concepteurs soient au courant de ce qui attend les logiciels de l'utilisateur.

Après ces définitions indispensables, quelle est la situation actuelle du texte dans les protocoles IETF ? La section 6 étudie la question. Il n'y a pas de règle générale, car certains protocoles IETF étaient internationalisés depuis le début (comme XMPP), d'autres ne l'ont été qu'après un long et pénible processus (cas du courrier électronique, créé sans penser à l'internationalisation et largement déployé, donc difficile à changer, lorsqu'on a voulu l'adapter à un monde vaste et varié).

Il y a donc encore du vocabulaire à apprendre. Ainsi, les **éléments de protocole** ("*protocol elements*") sont les éléments qui nomment une partie du protocole. Certains ne sont pas textuels, comme les numéros de port dans TCP, d'autres le sont (comme les commandes SMTP, par exemple `EHLO` et `MAIL FROM`) mais ne sont pas vus par l'utilisateur et donc, ne sont pas sujets à l'internationalisation (d'ailleurs, `EHLO` n'est pas de l'anglais non plus...).

Quant à l'**encodage sur le câble** ("*on-the-wire encoding*"), il désigne les bits qui passent effectivement sur le réseau. Cet encodage est souvent très différent <https://www.bortzmeyer.org/representation-texte.html> de la forme texte que verront les humains.

Mais il ne pose pas de problèmes d'internationalisation, au contraire du **texte analysé** ("*parsed text*") qui désigne le texte libre qu'un programme va analyser pour essayer d'y trouver des éléments de protocole. Le cas le plus connu est sans doute la convention d'ajouter un `Re:` (pour "*Reply*") dans le sujet d'un message. Un tel système est clairement très fragile, la localisation de ce préfixe suffisant à empêcher l'analyse. Il ne devrait pas apparaître dans les protocoles récents.

Lorsqu'un protocole transporte du texte, et que la connaissance de la langue dans laquelle est écrit ce texte est importante (par exemple pour déterminer l'ordre de classement), il faut utiliser une **identification de langue** ("*language identification*"). C'est un mécanisme par lequel on indique la langue utilisée, en

se servant des étiquettes de langue du RFC 5646. Ainsi, HTTP va utiliser l'en-tête `Content-Language: gsw-FR` dans la réponse, pour indiquer que le document servi est en alsacien <`https://www.bortzmeyer.org/enregistrement-alsacien.html`>.

Le texte contenu dans les échanges normalisés par un protocole IETF est souvent décrit par ASN.1. Ce langage permet plusieurs sortes de chaînes de caractère, de la traditionnelle `IA5String`, qui contient les caractères ASCII, à `UTF8String` qui, comme son nom l'indique, peut stocker tout Unicode. Ainsi, lorsque le RFC 5280 utilise ce type pour des champs d'un certificat X.509, il permet à ce format d'avoir des noms de CA ou d'utilisateurs complètement internationalisés.

Il existe également toute une terminologie spécifique aux IDN. On la trouve dans le RFC 5890 et résumée dans la section 7.

Plus nouveau est le problème des **variantes** ("*variants*"). Il date du RFC 3743 et peut se définir comme « des chaînes de caractères que certains humains, dans certaines conditions, considèrent comme équivalents ». Dans le RFC 3743, il ne s'agissait que de la différence entre les sinogrammes traditionnels et les sinogrammes simplifiés, pour lesquels une solution était recommandée dans le RFC 4713. Mais la question des variantes peut surgir dans bien d'autres cas :

— Des caractères que l'œil peut confondre comme le 1 et le l de l'alphabet latin, ou comme le a latin et le [Caractère Unicode non montré] cyrillique.
— Des caractères qui sont « le même » d'un certain point de vue, mais à qui Unicode a quand même affecté des numéros différents. Un exemple typique est le [Caractère Unicode non montré] grec, qui s'écrit [Caractère Unicode non montré] à la fin d'un mot, alors qu'il s'agit du même sigma. Autre exemple, le [Caractère Unicode non montré] arabe (le yeh), qui est codé différemment dans d'autres langues, par exemple [Caractère Unicode non montré] en persan.
— Et bien sûr les variantes orthographiques. Est-ce que "*color*" et "*colour*", ou bien "*theatre*" et "*theater*" sont le même mot en anglais ? Et "*strasse*" et "*stra[Caractère Unicode non montré]e*" en allemand ? Concrètement, un protocole Internet doit-il les considérer comme liés, comme faisant partie du même lot, l'un étant une variante de l'autre ?

La question des variantes a donc deux sous-questions : comment classer les caractères et les chaînes en lots (où les membres d'un lot sont considérés variantes l'un de l'autre) ? Et, surtout, le protocole doit-il intégrer cette notion, ou bien considérer plus traditionnellement que deux chaines de caractère sont différentes si elles ont des numéros Unicode différentes, point ?

Dernière section avec des définitions, la section 8 est consacrée aux termes divers. On y trouve, par exemple, le **profil local** ("*locale*"), qui est l'ensemble des informations qui varient selon le lieu ou la culture, et qui sont nécessaires pour que le programme, non seulement parle la langue de l'utilisateur, mais lui présente des informations comme la date ou la monnaie locale.

Les profils locaux sont une source de questions philosophiques sans fin. Par exemple, pour afficher un texte, faut-il utiliser le profil de l'auteur du texte ou bien celui du lecteur ? Et le profil doit-il dépendre de la personne, ou bien de sa localisation physique ? (Si je suis à Tokyo, je veux l'heure locale mais mes textes toujours en français.)

Autres termes utile, ceux de **translittération** ("*transliteration*"), de **transcription** ("*transcription*") et de **romanisation** ("*romanization*"). La translittération est la conversion d'une écriture dans une autre, en général lettre par lettre. Elle est réversible, la plupart du temps, puisque chaque lettre a été convertie. Cela marche bien entre écritures alpabétiques. La transcription, elle, suit les sons (et non pas les lettres) et va donc être différente selon la langue cible. C'est c'est ainsi que le nom du fondateur du parti bolchevique russe, [Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré], peut être translittéré Ul'ya[Caractère Unicode non montré

]nov (le résultat exact dépend de la norme suivie car il existe plusieurs normes de translittération de cet alphabet) mais est transcrit Ulyanov en anglais et Oulianov en français. Avec les écritures comme le chinois, la translittération n'est en général pas possible et il n'y a que des transcriptions, comme le pinyin. On nomme romanisation une conversion (translittération ou transcription, même si le RFC n'indique que la première possibilité) vers l'alphabet latin.

Quels sont les changements depuis le précédent RFC, le RFC 3536 ? L'annexe C les résume. Il y a des clarifications, pas de retraits mais plusieurs ajouts, notamment sur la question des variantes. La section 7, sur la terminologie spécifique aux IDN, est toute nouvelle.

Il existe bien sûr d'autres documents qui ont développé un vocabulaire cohérent sur l'internationalisation, et qui ont donc inspiré ce RFC. On peut citer la norme Unicode, le « "*Character Model for the World Wide Web*" <http://www.w3.org/TR/charmod/> » du W3C, et les RFC 2277, RFC 5646 et RFC 5890. Patrick Andries a un excellent glossaire de l'internationalisation en français <http://hapax.qc.ca/pdf/glossaire.pdf>.

L'annexe A contient une bibliographie sur le sujet de l'internationalisation, où on trouve des ouvrages de référence comme la somme sur les écritures du monde, « "*The world's writing systems*" <https://www.bortzmeyer.org/worlds-writing-systems.html> ». Sur le sujet d'Unicode, je recommande ces livres :

— « Unicode 5.0 en pratique <https://www.bortzmeyer.org/unicode-en-pratique.html> » de Patrick Andries
— « "*Unicode explained*" <https://www.bortzmeyer.org/unicode-explained.html> » de Jukka Korpela

Enfin, on trouve plein de choses en français sur le site de Patrick Andries <http://hapax.qc.ca/>.